GPU Parallel Implementation of Dual-Depth Sparse Probabilistic Latent Semantic Analysis for Hyperspectral Unmixing

José Antonio Gallardo, Mercedes E. Paoletti, *Student Member, IEEE,* Juan M. Haut, *Student Member, IEEE,* Ruben Fernandez-Beltran, Antonio Plaza, *Fellow, IEEE,* and Javier Plaza, *Senior Member, IEEE*

***Abstract*—Hyperspectral unmixing (HU) is an important task for remotely sensed hyperspectral (HS) data exploitation. It comprises the identification of pure spectral signatures (endmembers) and their corresponding fractional abundances in each pixel of the HS data cube. Several methods have been developed for (semi-) supervised and automatic identification of endmembers and abundances. Recently, the statistical dual-depth sparse probabilistic latent semantic analysis (DEpLSA) method has been developed to tackle the HU problem as a latent topic-based approach in which both endmembers and abundances can be simultaneously estimated according to the semantics encapsulated by the latent topic space. However, statistical models usually lead to computationally demanding algorithms and the computational time of DEpLSA is often too high for practical use, in particular when the dimensionality of the HS data cube is large. In order to mitigate this limitation, this paper resorts to graphical processing units (GPUs) to provide a new parallel version of DEpLSA, developed using the NVidia Compute Device Unified Architecture (CUDA). Our experimental results, conducted using four well-known HS datasets and two different GPU architectures (GTX 1080 and Tesla P100) show that our parallel versions of DEpLSA and the traditional pLSA approach can provide accurate HU results fast enough for practical use, accelerating the corresponding serial versions in at least 30x in the GTX 1080 and up to 147x in the Tesla P100 GPU, which are quite significant acceleration factors that increase with image size, thus allowing for the possibility of fast processing of massive HS data repositories.**

***Index Terms*—Graphics Processing Unit (GPU), Hyperspectral Unmixing (HU), probabilistic generative models, probabilistic Latent Semantic Analysis (pLSA). Dual-Depth Sparse pLSA (DEpLSA).**

This work has been supported by the Spanish Education Ministry (FPU14/02012, FPU15/02090), EU FEDER (ESP2016-79503-C2-2-P) and the Spanish MINECO (TIN 2015-65277-R) and Generalitat Valenciana (APOSTD/2017/007). This work has also been supported by Junta de Extremadura (Decreto 14/2018, de 6 de febrero, por el que se establecen las bases reguladoras de las ayudas para la realizacin de actividades de investigacin y desarrollo tecnolgico, de divulgacin y de transferencia de conocimiento por los Grupos de Investigacin de Extremadura, Ref. GR18060) and the European Union's Horizon 2020 research and innovation programme under grant agreement No. 734541 (EOXPOSURE).

J. A. Gallardo, M. E. Paoletti, J. M. Haut, J. Plaza and A. Plaza are with the Hyperspectral Computing Laboratory, Department of Technology of Computers and Communications, Escuela Politécnica, University of Extremadura, PC-10003 Cáceres, Spain. (e-mail: jgallardst@alumnos.unex.es; mpaoletti@unex.es; juanmariohaut@unex.es; aplaza@unex.es; jplaza@unex.es). R. Fernandez-Beltran is with the Institute of New Imaging Technologies, University Jaume I, 12071 Castellón de la Plana, Spain. (e-mail: rufernan@uji.es).

I. Introduction

Over the past years, hyperspectral (HS) imaging has shown to be an excellent tool to deal with many different remote sensing problems [1], [2]. From detailed Earth surface classification [3]–[5], through fine-grained land cover mapping [6], [7], to precise material identification and analysis [8], [9], there are multiple domains within the remote sensing field where the spectral-spatial precision of air-borne and space-borne HS data becomes particularly useful. In particular, one of the most relevant research areas to uncover sub-pixel information from HS images is the so-called Hyperspectral unmixing (HU) task [10], [11]. Specifically, HU pursues the objective of decomposing a HS remotely sensed scene into two main constitutive components: (i) endmembers and (ii) abundances. On the one hand, endmembers represent the spectral signatures of the most spectrally pure components contained in the scene. On the other hand, fractional abundances provide the corresponding amount of each spectrally pure component that is present at each image pixel.

In the literature, extensive research work has been conducted to effectively deal with the ill-posed nature of the HU problem [10]. One of the most popular types of HU techniques is the geometrical approach, which makes use of the own data geometry to estimate both endmembers and abundances. In this regard, the vertex component analysis (VCA) [12] considers that spectral signatures describe a minimum volume simplex that contains the data, hence the HU task can be efficiently carried out using the convex geometry discipline. Other geometrical methods, such as the minimum volume simplex analysis (MVSA) [13], introduce some additional constraints on this convex scheme to improve the model robustness. Another relevant group of HU techniques is the statistical approach. More specifically, this kind of methods deal with the unmixing problem considering endmembers and abundances as probability distributions. In the literature, it is possible to find different statistical methods, such as [14] and [15] which model the HU task using Dirichlet and Gaussian distributions, respectively. Additionally, there are other unmixing techniques available that cope with the HU problem from a matrix decomposition perspective, such as the non-negative matrix factorization (NMF) [16] and the robust collaborative non-negative matrix factorization (R-CoNMF) [17].

To some extent, all these methodologies have shown to be effective to unmix HS remote sensing data under specific conditions [11]. Whereas geometrical approaches struggle at uncovering spectral signatures on highly mixed scenarios, statistical and decomposition techniques provide a more powerful HU scheme since the HS data can be managed from a more general perspective [10]. Furthermore, some recent research lines show the advantages of using the so-called semantic representations when processing HS data [18], being probabilistic topic models an emerging statistical technology within the remote sensing field [19]–[21]. In general, topic models are a kind of probabilistic generative models that become particularly useful to represent visual data at a higher abstraction level by means of their hidden semantic patterns [22]. As a result, these models have been recently used to uncover complex spectral relationships while providing competitive advantages in the HU domain [23].

More specifically, the work presented in [23] defines a novel probabilistic topic model, called Dual-Depth Sparse probabilistic Semantic Analysis (DEpLSA) – inspired by the traditional pLSA [24]– which is specifically designed to effectively uncover spectral signatures and fractional abundances from real HS remotely sensed data. In fact, this seminal work shows the potential of probabilistic generative models and also the advantages

of DEpLSA with respect other state-of-the-art unmixing techniques. However, there is a key factor that may limit its practical usage in actual remote sensing operational environments: the computational cost. Note that probabilistic generative models, in general, and DEpLSA, in particular, have a high computational complexity due to the NP-complete nature of the Bayesian learning process [25], [26]. As a result, more research work is still required to study the viability of integrating these kinds of procedures in actual remote sensing production enviroments.

Despite the fact that some works in the literature try to exploit different parallel techniques for some related probabilistic generative architectures [24], [27], [28], the specific DEpLSA nature together with the especial complexity of the HU field generate particular demands that cannot be addressed from a general purpose perspective. Concretely, the advances in the systems used to capture hyperspectral images have increased their complexity. Such complexity makes traditional methods based on single and multi-core CPUs outdated, as they cannot cope with the required computational needs in order to process large volumes of data. In this situation, our implementation becomes a reliable alternative, capable of processing large volumes of data in a reasonable amount of time. Note that processing remotely sensed data using parallel architectures faces some technical challenges which are not present in other fields [29], besides the inherent spatial-spectral intricacy of the HS domain make necessary to develop and test target-based efficient implementations. Precisely, this is the gap that motivates this work.

In this scenario, the work presented here proposes a new graphic processing unit (GPU)-based parallel implementation of the HU method defined in [23], in order to enable the use of the newly DEpLSA unmixig model in actual operational environments of different Earth Observation programs and missions. Specifically, we take advantage of the Expectation-Maximization (EM) optimization algorithm employed in [23] to integrate different parallel optimizations based on the Compute Unified Device Architecture (CUDA)[1] platform for GPU hardware devices. Our work is largely driven by the success of several available CUDA implementations of HS processing algorithms on GPU devices. For instance, in [30], an automatic target detection and classification algorithm is accelerated. In [31], a highly parallel GPU architecture for lossy hyperspectral image compression is presented. The work in [32] presents a multi-GPU implementation of the MVSA algorithm for spectral unmixing purposes. A massively parallel GPU design is discussed in [33] for target detection purposes. Other advanced algorithms for HS data exploitation have been successfully accelerated on GPUs using the CUDA architecture, including composite kernels [34], iterative-constrained endmember extraction [35], support vector machines [36], real-time unmixing [37], [38], HS subspace identification [39], spatial-spectral preprocessing [40], segmentation [41], linear unmixing chains [42], isometric mapping [43], registration [44] or spatially adaptive classification [45], among many others [46]. Note the wide acceptance of GPU-based implementations of HS unmixing algorithms [47], which led us to consider GPUs as a potentially efficient solution for accelerating our DEpLSA algorithm.

In the experimental part of the work, we compare the proposed GPU DEpLSA implementation for HS unmixing purposes with a baseline single-core version and also a parallel multi-core implementation of the DEpLSA model. The obtained quantitative and qualitative results, using four real HS datasets, reveal the performance advantages of the proposed approach for real-life remote

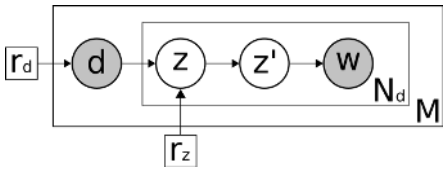
[1]https://developer.nvidia.com/cuda-zone

sensing production chains.

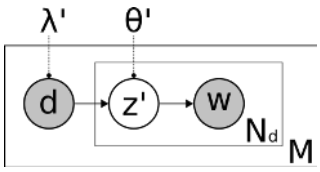
The remainder of the paper is organized as follows. Section II describes the background behind the DEpLSA unmixing model. Section III presents in detail the proposed GPU-based parallel implementation. Section IV provides the experimental results and discussion. Finally, section V concludes the work with some remarks and hints at plausible future research lines.

II. HU DEpLSA-BASED MODEL

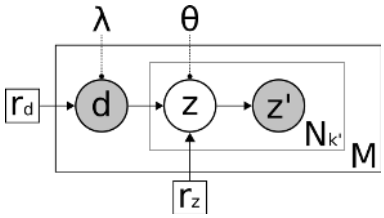
The DEpLSA approach [23] can be considered a statistical HU method based on the concept of latent topics [48], where the unmixing problem is faced as a latent topic-based approach, aiming at estimating endmembers and their corresponding fractional abundances, according to the semantics encapsulated by the latent topic space. In particular, it defines a semi-generative HU model by considering two latent context variables, i.e. z and z', associated to different abstraction levels when conducting the unmixing process over the input HS image. As it can be ssen in the DEpLSA model graphical representation (Fig. 1a), image pixels are represented by the observable random variable d, the dual-hierarchy of spectral patterns are described by the hidden variables z' (deep-topics, used to generate the semantic representation of the input spectral data) and z (restricted-topics, used to learn endmembers and abundances in the semantic space), and the input pixel spectra are encapsulated by the observable random variable w. In addition, M is the total number of input pixels and N_d represents the number of reflectance activations within each pixel spectra. Considering that r_d and r_z are two diverging regularization factors to guarantee a certain sparsity constraint, fractional abundances are described by the conditional probability $p(z|d)$ and spectral signatures correspond to the $p(w|z)$ probability distribution.



(a) DEpLSA



(b) DEpLSA-1



(c) DEpLSA-2

Fig. 1. Original DEpLSA model (a) and two-phase model relaxation (b)-(c).

From a practical point of view, the main advantage of DEpLSA unmixing model is the utilization on the deep-topic space (z') to generate a high-dimensional semantic characterization of the original data using K' components. Then, the restricted-topics (z) are applied to effectively infer the K endmembers and the corresponding fractional abundance maps over this semantic space. However, this dual-depth architecture implies an important computational cost since an additional degree of freedom is introduced when capturing the relationships between z and z' random variables. Therefore, it is necessary to apply the DEpLSA unmixing model using the following two-step model relaxation:

- **DEpLSA-1** (Fig. 1b) where the deep-topic probability distributions with K' components, $\lambda' \sim p(z'|d)$ and $\theta' \sim p(w|z')$, are estimated using the input HS data.
- **DEpLSA-2** (Fig. 1c) where the deep-topic ran-

dom variable (z') becomes observable being approximated by the previous λ' distribution. In this way, the fractional abundances can be inferred as $\lambda' \sim p(z'|d)$ and the K spectral signatures can be computed using both θ' and θ.

Note that this model relaxation reduces the original DEpLSA unmixing model complexity since the dual-hierarchy of patterns is unfolded in two sequential steps by assuming an uniform prior probability over deep-topics. Specifically, both steps are estimated by maximizing the complete log-likelihood using the EM algorithm [49]. After applying the Jensen's inequality to the log-likelihood term, inserting the appropriate Lagrange multipliers, computing the partial derivatives and isolating the corresponding model parameters, it is possible to derive the following equations for the EM-based optimization,

$$p(z'|w,d) = \frac{p(w|z')p(z'|d)}{\sum_{z'} p(w|z')p(z'|d)} \tag{1}$$

$$\theta' \sim p(w|z') = \frac{\sum_{d} n(w,d)p(d)p(z'|w,d)}{\sum_{w}\sum_{d} n(w,d)p(d)p(z'|w,d)} \tag{2}$$

$$\lambda' \sim p(z'|d) = \frac{\sum_{w} n(w,d)p(z'|w,d)}{\sum_{z'}\sum_{w} n(w,d)p(z'|w,d)} \tag{3}$$

$$p(z|z',d) = \frac{p(z'|z)p(z|d)}{\sum_{z} p(z'|z)p(z|d)} \tag{4}$$

$$\theta \sim p(z'|z) = \frac{\sum_{d} n(z',d)p(d)p(z|z',d) - \delta_z/K'}{\sum_{z'}\sum_{d} n(z',d)p(d)p(z|z',d)} \tag{5}$$

$$\lambda \sim p(z|d) = \frac{\sum_{z'} n(z',d)p(z|z',d) - \delta_d/K}{\sum_{z}\sum_{z'} n(z',d)p(z|z',d)} \tag{6}$$

where Eqs. (1)-(3) correspond to the E-step and M-step of DEpLSA-1, and Eqs. (4)-(6) are the ones for DEpLSA-2. Additionally, K is the number of endmembers, K' represents the number of component of the deep-topic space ($K' >> K$), $n(w,d)$ are the original reflectance pixel activations and $n(z',d)$ is approximated by λ'. Regarding the EM procedure itself, it is performed as follows. Initially, the corresponding model parameters are initialized. Then, E-step and M-step are alternated until the model converges, whether using a 10^{-6} stability threshold in log-likelihood or a maximum of 10^3 EM iterations. Algorithms 1-2 show a more detailed description of the procedures, summarizing their main computations.

Algorithm 1 EM-based proceduce for DEpLSA-1

Input $n(w,d)$: Input reflectance pixel activations

Input K': High-dimensional semantic space components

Output θ': $p(w|z')$

Output λ': $p(z'|d)$

procedure DEPLSA1($n(w,d)$, K')
  $I = 0$
  $T = \infty$
  $L = 0$
  $\lambda' \leftarrow$ Random initialization
  $\theta' \leftarrow$ Random initialization
  while $(I < 10^3)$ & $(T > 10^{-6})$ **do**
    $p(z'|w,d) \leftarrow$ Eq.1
    $p(w|z') \leftarrow$ Eq.2
    $p(z'|d) \leftarrow$ Eq.3
    $\ell_c \leftarrow$ Compute log-likelihood
    $T = \ell_c - L$
    $L = \ell_c$
    $I++$
  end while
end procedure

After DEpLSA-1 and DEpLSA-2 models have been

Algorithm 2 EM-based proceduce for DEpLSA-2

Input $n(z', d)$: λ'
Input K: Number of endmembers
Input r_d: Sparsity constraint for d
Input r_z: Sparsity constraint for z
Output θ: $p(z'|z)$
Output λ: $p(z|d)$

procedure DEPLSA2($n(z', d)$, K, r_d, r_z)
  $I = 0$
  $T = \infty$
  $L = 0$
  $\lambda' \leftarrow$ Uniform initialization
  $\theta' \leftarrow$ Random initialization
  while $(I < 10^3)$ & $(T > 10^{-6})$ **do**
    $p(z|z', d) \leftarrow$ Eq.4
    $p(z'|z) \leftarrow$ Eq.5
    $p(z|d) \leftarrow$ Eq.6
    $\ell_c \leftarrow$ Compute log-likelihood
    $T = \ell_c - L$
    $L = \ell_c$
    $I++$
  end while
end procedure

sequentially applied and successfully converged, the final estimation for the fractional abundances corresponds to parameter $\lambda \sim p(z|d)$ and the endmembers can be factorized as shown in Eq. (7).

$$p(w|z) = \sum_{z'} \overbrace{p(w|z')}^{\text{DEpLSA-1}} \overbrace{p(z'|z)}^{\text{DEpLSA-2}} = \Theta'\Theta. \quad (7)$$

III. GPU Parallel Implementation for Hyperspectral Unmixing Based on CUDA

In this section we provide a detailed description of the developed parallel implementation of proposed algorithm. In particular, we will focus on providing a parallel implementation of the most time consuming operations of the DEpLSA algorithm. The memory allocation and I/O transfer between the host (CPU) and the devices (GPU) will also be optimized.

In this context, we will focus on EM algorithm which, as mentioned above, can be considered as the basis of dpLSA algorithm and represents its most computationally intensive part. All the operations of this algorithm are computations on probability matrices and, therefore, a simple yet efficient strategy to parallelize this algorithm is to partition matrix operations across different cores of a many-core device, which will also enable the redistribution of workloads at execution time. Such runtime redistributions are possible thanks to the way CUDA manages the computing *threads*. Specifically, CUDA creates a two-layer hierarchy, where the first one contains a grid that holds a per-kernel fixed number of blocks in a one-dimensional (1-D), 2-D or 3-D way. Inside of each block, there is a pool of threads whose dimensionality can also be from one to three dimensions; such dimensionality is also parametrized per-kernel. Since those dimensions are parameters of each kernel call, they can be adjusted to fit the output matrix dimensionality, guaranteeing per-thread complete atomicity. A visual example of the hierarchical strategy adopted by CUDA to manage threads is provided in Fig. 2

A. Optimization of the Memory Allocation and I/O Transfer

In DEpLSA, the data computed across the EM algorithm is stored inside three matrices: θ (endmembers), λ (abundances), and the original pixel vectors. These matrices need to be allocated inside the GPU (device). In this regard, there are two possibilities: i) making constant input/output (I/O) transfers by holding only the necessary matrices inside the device memory, or ii) storing all data in video memory across the entire computing process.

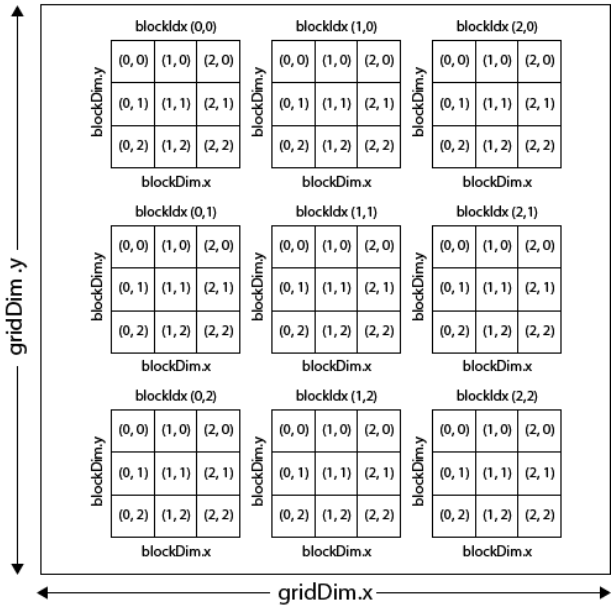


Fig. 2. Graphical illustration of CUDA 2-D grid and block hierarchy.

While the first alternative is intended to optimize memory management in massive data scenarios, it can suffer from significant bottlenecks as a result of massive data transfers, so strategy ii) has been adopted in order to minimize the transfer time in our implementation.

It is also important to emphasize that our implementation may face challenges when handling extremely large hyperspectral images that need to be stored in the device (GPU) memory. Alternatively, there are some techniques that can alleviate this situation, e.g. by keeping the I/0 transfers constant during the analysis. This sacrifices some efficiency in terms of time, but also allows larger data sets to be processed. Specifically, this can be done by storing in device memory just the matrices that are strictly required for the actual step executed by the kernel. Another possibility is to use a batch-based procedure, in which each iteration is split in terms of data and only a subset of pixels are loaded in memory and processed at a given moment. As said before, all these methods also have a cost in terms of performance.

B. Parallel Optimization of the Expectation Step

As explained above, the main goal of this step is to generate a new probabilistic latent space, which is computed based on the actual probabilities carried out by the matrices λ (abundances) and θ (endmembers) and stored into a 3-D structure called p, as shown in Eq. 1. Since this structure conveys the computing results, atomicity over each index needs to be guarantee. To achieve this, the kernel's dimensions are set to ensure each thread is in charge of processing always the same p value and store denominators in the per-block shared memory, as Fig. 3 shows. Algorithm 3 shows the pseudocode of our parallel implementation of the Expectation step. As it can be seen, the $thread$ index references the value of matrices processed by this particular thread, and $block$ index references the per-block shared denominator.

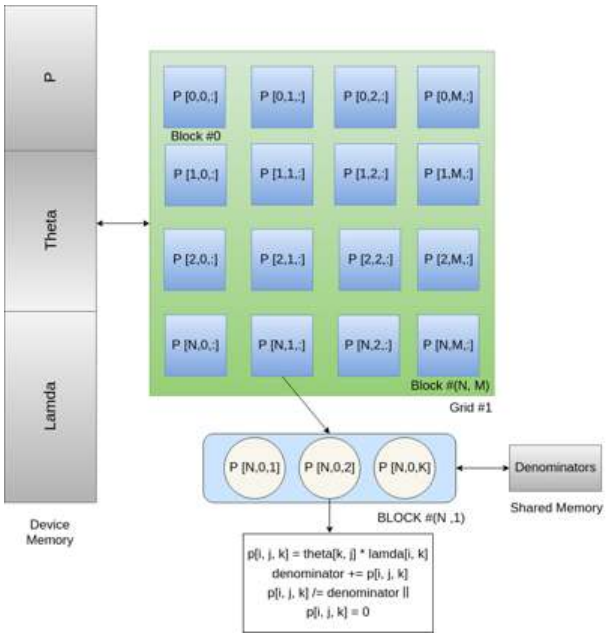


Fig. 3. Device state while executing the kernel corresponding to the Expectation step. Grid hierarchy and memory states are shown in this diagram.

Algorithm 3 Expectation step kernel

```
1: procedure KERNEL
2:     for Block in Grid[X,Y] do                ▷ In parallel
3:         den ← 0                              ▷ Per-block shared
4:         for Thread in Block[Z] do            ▷ In parallel
5:             P[thread] ← λ[thread] × θ[thread]
6:             den[block] ← den[block] + P[thread]   ▷ Atomic
7:             P[thread] ← P[thread]/den[block]
8:         end for
9:     end for
10: end procedure
```

As seen above, the parallelization of the Expectation step highly relies on computing each value of P matrix in parallel. In order to achieve this task, we use a simple kernel structure that relies on the per-block shared memory to handle the common block denominators that will divide the per-core computed value of the P, based on λ and θ. This shared value is atomically increased and computed as the sum of computed core, P.

C. Parallel Optimization of the Maximization Step

Instead of a single step in the sequenatial implementation, our CUDA implementation of the maximization step partitions the entire process into a subset of kernels in order to change the grid dimensions as needed to preserve atomicity at runtime.

First, the endmember matrix (θ) is updated by chaining a subset of kernels, dividing Eq. (2) into three main steps:

1) The first step updates the fraction numerator (this is performed by the kernel described in Algorithm 4). As this value is computed using the full pixel information, and the number of pixels exceeds the maximum number of per-block cores, there needs to be a `for` loop inside the kernel in order to compute $theta$.
2) The second step performs the sums on the denominator (this is accomplished by the kernel in Algorithm 5). This kernel just computes the denominator as a subset of the per-column θ values.
3) The last step performs the division and assigns it into θ (this is done by the kernel in Algorithm 6). The last step of this process consists of dividing the outputs of the first two kernels atomically into each core.

In this case, the dependencies among the operands of the denominator summatory happen above block-level, thus making the use of shared memory impossible. In addition, the block dimension is directly related to the total amount of pixels, which is greater than the maximum number of available threads per block that can be allocated (1024). Therefore, a for-loop inside the kernel is needed, which has a slight effect on the final performance of the algorithm. For illustrative purposes, Fig. 4 shows an overview diagram illustrating this process.

Algorithm 4 Endmembers (θ) numerator computing kernel

procedure KERNEL
  for `Block` in `Grid[M,K]` **do** ▷ In parallel
    for `Thread` in `Block[1024]` **do** ▷ In parallel
      $\theta[block] \leftarrow 0$
      for `step` in `steps]` **do**
        $aux \leftarrow (X[step] \times P[step]) - regularizer$
        if $aux > 0$ **then**
          $\theta[block] \leftarrow aux$ ▷ Atomic
        end if
      end for
    end for
end procedure

Algorithm 5 Endmembers (θ) denominator computing kernel

procedure KERNEL
  for `Block` in `Grid[K]` **do** ▷ In parallel
    for `Thread` in `Block[M]` **do** ▷ In parallel
      $den[block.z] \leftarrow den[block.z] + \theta[thread]$
    end for
  end for
end procedure

Algorithm 6 Endmembers (θ) division computing kernel

procedure KERNEL
  for `Block` in `Grid[M, K]` **do** ▷ In parallel
    if $den[block.z] \neq 0$ **then**
      $\theta[thread] \leftarrow \theta[thread]/den[block.z]$
    else
      $\theta[thread] \leftarrow 1/M$
    end if
  end for
end procedure

Algorithm 7 Abundances (λ) computing kernel

procedure KERNEL
  for `Block` in `Grid[X,Z]` **do** ▷ In parallel
    $den \leftarrow 0$ ▷ Per-block shared
    for `Thread` in `Block[Y]` **do** ▷ In parallel
      $aux \leftarrow (X[step] \times P[step]) - regularizer$
      if $aux > 0$ **then**
        $\lambda[block] \leftarrow aux$ ▷ Atomic
      end if
      $den \leftarrow den + X[thread]$ ▷ Atomic
      if $den \neq 0$ **then**
        $\lambda[thread] \leftarrow \theta[thread]/den$
      else
        $\lambda[thread] \leftarrow 1/K$
      end if
    end for
  end for
end procedure

After the endmember-related computations are completed, the Maximization step tries to find the best abundances from the latent space computed in the Expectation step, in a very similar way as the calculation for the endmembers. However, as the kernel block size in charge of computing the abundances depends on the

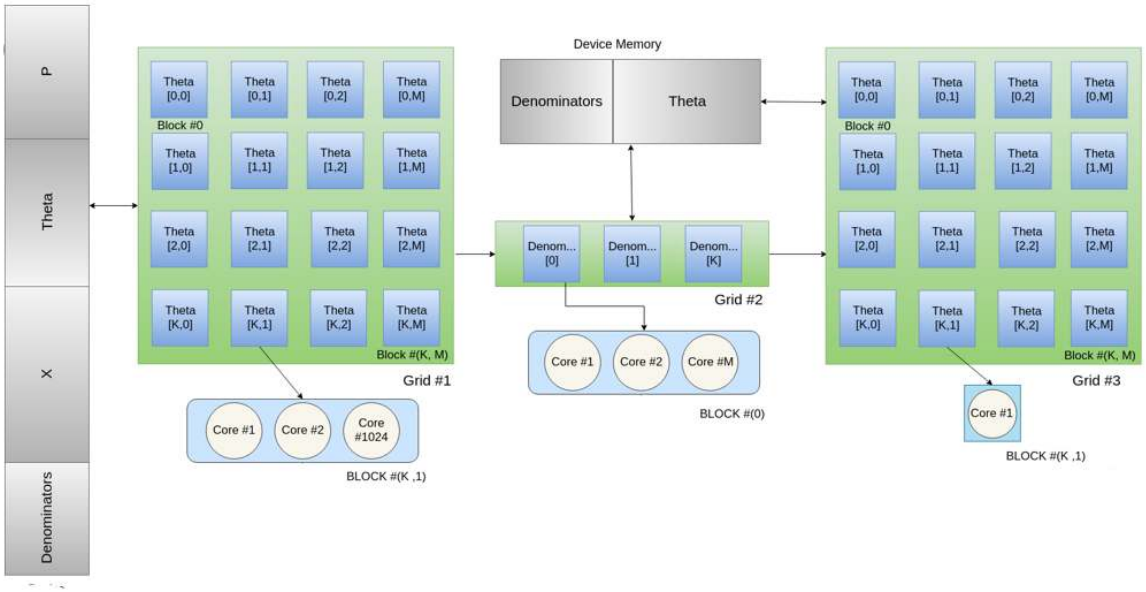


Fig. 4. Full pipeline describing the endmember-related computations on the Maximization step. This diagram covers the entire process for computing the values of the endmember matrix (θ) in Grid #3 by dividing the returned values from Grids #1 and #2.

number of bands of the input image, it is easier to ensure atomicity in this case, creating a kernel stack that performs the calculation of the whole Eq. (3). In this case, each block is considered as a matrix with dimensions $N \times K$, containing a vector of M threads per block, as shown in Fig. 5. A pseudocode for the kernel that implements this step is given in Algorithm 7. As it can be seen, the operations are similar to those performed by Algorithms 4-6. Here, as it was already the case for the computation of the Expectation kernel, we rely on the per-block shared memory to compute the new abundances. A subset of image pixels is used to divide the newly computed λ values among the cores, based on the iteration latent space. .

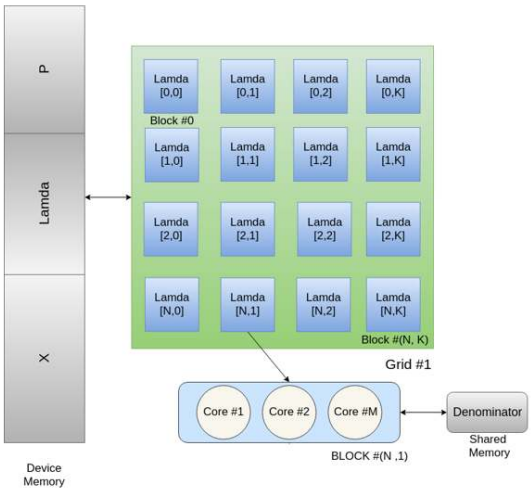


Fig. 5. Full pipeline describing the abundance-related computations on the Maximization. This diagram covers the entire process from computing the λ values based on the input image data and the predicted latent space from the Expectation step.

IV. Experiments

A. Enviroment

In order to evaluate the computational performance of the DEpLSA-GPU implementation (and also of a

GPU implementation of the traditional pLSA), serial versions (that will be used as a baseline for the speedup calculations) have been implemented and executed in a host hardware environment with a 6th Generation Intel® Core™i7-6700K processor with 8M of Cache and up to 4.20GHz (4 cores/8 way multi-task processing), installed over an ASUS Z170 pro-gaming motherboard. The available memory is divided into 40GB of DDR4 RAM with a serial speed of 2400MHz and a Toshiba DT01ACA HDD with 7200RPM and 2TB of storage capacity. The paralellel implementations of pLSA-GPU and DDpLSA-GPU have been executed in two different GPUs:

1) An NVIDIA GeForce GTX 1080, composed by 2560 CUDA cores, with 8GB GDDR5X of video memory and 10 Gbps of memory frequency (referred to hereinafter as GPU1).
2) A Tesla P100 GPU, with 3584 CUDA cores, 16GB HBM2 video memory and 12 Gbps of memory frequency (referred to hereinafter as GPU2).

In order to compare our GPU versions with a common CPU implementation, experiments have been conducted against the serial baselines, which run on top of a C++ library that allows tensor work called *xtensor*. This library optimizes all matrix-related computations and assignment tasks. On this version, the kernels in Algorithms 3-7 are implemented in a very similar way, being the only difference that the tasks does not run in parallel.

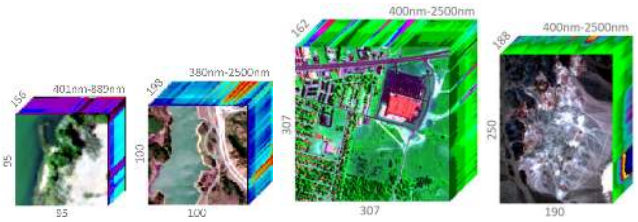
Two different serial versions have been carried out, both of them compiled with the GNU C++ (g++) compiler. The first one is a pure serial version, without any kind of optimization and can be considered as the baseline implementation, while the second one has been compiled using $-O3$ and $-xAVX$ in order to provide automatic vectorization. We refer to this optimized version hereinafter as OP-DEpLSA (with the optimized pLSA-based version being referred to as OP-pLSA). By running experiments against this full set of versions, we are able to provide results for non-parallel, data-parallel and massively-parallel versions of our algorithms.

B. Datasets

In this work, the following four real hyperspectral images have been used in the experimental validation (Fig. 6):

- Samson (Fig. 6.a) [50] is a popular hyperspectral dataset which contains 952×952 pixels and 156 bands, ranging from 380 nm to 2500 nm wavelengths. In particular, a region of interest with 95×95 pixels has been selected from the (252,332)-th coordinate, resulting in a final size of $95 \times 95 \times 156$. The Samson image includes three different endmembers: soil, tree and water.
- Jasper Ridge (Fig. 6.b) [50] is another common hyperspectral image with 512×614 pixels and 224 channels, covering the spectral range from 380 nm to 2500 nm. Specifically, we have considered a region of 100×100 pixels starting from the (105,269)-th coordinate. Additionally, channels 1-3, 108-112, 154-166 and 220-224 have been removed due to atmospheric effects, obtaining a final size of $100 \times 100 \times 198$. The Jasper dataset contains four different spectrally pure signatures: road, soil, water and tree.
- Urban (Fig. 6.c) [50] is hyperspectral dataset which comprises 307×307 pixels and a total of 210 bands from the 400 nm to the 2500 nm wavelength. In order to avoid atmospheric effects, bands 1-4, 76, 87, 101-111, 136-153 and 198-210 bands have been removed, obtaining a final size of $307 \times 307 \times 162$. The considered Urban scene includes four different pure materials: asphalt, grass, tree and roof.

- Cuprite (Fig. 6.d) [50] is probably one of the most popular and challenging images in the area of hyperspectral unmixing. The original dataset contains 224 spectral channels. However, a total of 188 bands have been considered in this work, after removing the noisy channels (1-2 and 221-224) and the water absorption ones (104-113 and 148-167). In addition, the considered region of interest includes 250×190 pixels, for a final size of $250 \times 190 \times 188$. The number of endmembers in the considered region of interest is twelve: Alunite, Andradite, Buddingtonite, Dumortierite, Kaolinite1, Kaolinite2, Muscovite, Montmorillonite, Nontronite, Pyrope, Sphene and Chalcedony.



(a) Samson (b) Jasper Ridge (c) Urban (d) Cuprite

Fig. 6. Hyperspectral datasets considered in the experiments.

C. Experimental Assessment

In order to asses and quantify the accuracy of the proposed hyperspectral unmixing technique, two different widely adopted metrics have been considered: Spectral Angle Distance (SAD) and Root Mean Squared Error (RMSE). Whereas SAD (Eq. 8) aims at quantitatively asses the K spectral signatures by computing the average spectral angle between the estimated endmembers ($\widetilde{\theta}$) and the ground-truth ones (θ), RMSE (Eq. 9) evaluates the quality of the fractional abundance maps by calculating the absolute difference between the estimated abundances ($\widetilde{\lambda}$) and the ground-truth ones (λ).

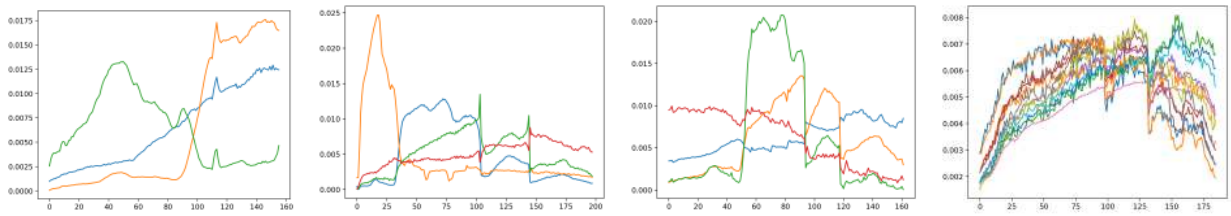
$$\mathrm{SAD}(\widetilde{\theta}, \theta) = \frac{1}{K} \sum_{i}^{K} \arccos \frac{\widetilde{\boldsymbol{\theta}}_i \cdot \boldsymbol{\theta}_i}{||\widetilde{\boldsymbol{\theta}}_i||\ ||\boldsymbol{\theta}_i||}. \quad (8)$$

$$\mathrm{RMSE}(\widetilde{\lambda}, \lambda) = \sqrt{\frac{1}{M} \sum_{i}^{M} (\widetilde{\lambda}_i - \lambda_i)^2}. \quad (9)$$

D. Results and Discussion

In this subsection we evaluate the performance of our implementations from the viewpoing of both unmixing accuracy and computational performance. Fig. 7 shows the obtained spectral signatures of the endmembers in the four considered datasets, employing the proposed method. These signatures will be considered as the ground-truth endmembers (θ) in the SAD calculations, while their corresponding abundance maps (λ) will be used as the ground-truth abundance maps for the RMSE calculations.

Table I shows the SAD-based and RMSE-based scores obtained after comparing the true versus estimated endmembers and abundance maps for each considered scene, respectively. For each dataset, we report the scores obtained by the original versions (pLSA, DEpLSA) and the GPU implementations (GPUpLSA, GPUDEpLSA). As it can be seen in the table, the value of the metrics depends on the complexity of the scenes (given by the number of endmembers K). In all cases, the SAD and RMSE values obtained by the original methods and their corresponding GPU versions is very similar, which indicates that the GPU versions provide almost the same results as the original counterparts. It should be noted that, for the Cuprite scene, the RMSE scores could not be computed as this scene only has ground-truth endmembers available (obtained from the well-known USGS library of mineral signatures), but there are no ground-truth fractional abundance maps that can be used for the calculation of the RMSE scores in this particular case.



(a) Samson (b) Jasper Ridge (c) Urban (d) Cuprite

Fig. 7. Obtained spectral signatures of the available endmembers in the four considered datasets: (a) Samson, (b) Jasper Ridge, (c) Urban and (d) Cuprite.

TABLE I

ACCURACY EVALUATION OF THE SERIAL AND PARALLEL VERSIONS OF PLSA AND DEPLSA IN TERMS OF SPECTRAL ANGLE DISTANCE (SAD) AND ROOT MEAN SQUARED ERROR (RMSE) ABUNDANCE ASSESSMENT (DIFFERENT DATASETS ARE SHOWN IN ROWS AND UNMIXING METHODS IN COLUMNS).

Datasets		Members (K)	Spectral Angle Distance - SAD ($\times 10^{-2}$)				Root Mean Squared Error - RMSE ($\times 10^{-2}$)			
			(A) State of art methods		(B) GPU parallel versions		(C) State of art methods		(D) GPU parallel versions	
			pLSA	DEpLSA	GPUpLSA	GPUDEpLSA	pLSA	DEpLSA	GPUpLSA	GPUDEpLSA
Real data	Samson	3	19.27 ±13.1	4.27 ±1.09	12.72 ±0.39	5.22 ±0.54	19.51 ±6.06	5.49 ±1.83	16.12 ±0.65	5.23 ±0.50
	Jasper	4	30.41 ±4.68	15.23 ±13.06	32.99 ±1.15	17.55 ±2.36	20.36 ±5.20	15.56 ±4.640	19.85 ±1.03	14.82 ±1.59
	Urban	4	33.24 ±18.83	13.84 ±13.41	37.56 ±5.05	14.18 ±1.49	17.48 ±4.79	13.65 ±4.69	18.00 ±1.50	11.64 ±0.76
	Cuprite	12	44.57 ±33.56	20.02 ±31.54	45.08 ±1.18	26.71 ±2.03	-	-	-	-

TABLE II

EXECUTION TIMES (IN SECONDS) FOR THE SERIAL (WITH AND WITHOUT OPTIMIZATION FLAGS) AND PARALLEL (EXECUTED ON THE TWO CONSIDERED GPUS) VERSIONS OF PLSA AND DEPLSA (DIFFERENT DATASETS ARE SHOWN IN ROWS).

Dataset	K	pLSA	OP-pLSA	GPU1-pLSA	GPU2-pLSA	Speedup Optimized	Speedup GPU1	Speedup GPU2
Samson	3	160.43	105.95	4.70	2.94	1.51	34.14	54.61
Jasper	4	264.44	198.87	8.15	4.61	1.33	32.45	57.43
Urban	4	3167.61	2643.43	88.96	39.76	1.20	35.61	79.68
Cuprite	12	5584.27	4622.68	163.66	55.51	1.21	34.12	100.60
Dataset	K	DEpLSA	OP-DEpLSA	GPU1-DEpLSA	GPU2-DEpLSA	Speedup Optimized	Speedup GPU1	Speedup GPU2
Samson	3	2440.85	3404.03	61.23	21.89	1.03	39.86	111.51
Jasper	4	2377.21	3331.28	90.09	31.17	1.02	37.78	109.21
Urban	4	41282.36	40158.00	-	280.01	1.03	-	147.43
Cuprite	12	26990.26	26791.35	-	198.68	1.01	-	135.85

For illustrative purposes, Fig. 9 shows the abundace maps and the absolute distance scores obtained for one particular dataset: the Samson scene in Fig. 6(a). Specifically, Figs. 9(a)-(c) show the ground truth abundances corresponding to the three endmembers in Fig. 7(a). Figs. 9(d)-(f) show the fractional abundance maps obtained by the DEpLSA algorithm (executed on the Tesla P100 GPU). Finally, Figs. 9(g)-(i) show the absolute distance between the estimated and real abundances for each of the three considered endmembers, where dark colors indicate lower errors. As it can be seen, the distances between the true and estimated abundances are very low, being demonstrated quantitatively in the Table I where the RMSE scores are also very low, indicating that the DEpLSA algorithm (executed in the GPU) does a very good job in the task of estimating abundances that are

TABLE III

PER-KERNEL EXECUTION TIME FOR BOTH PLSA AND DEPLSA IMPLEMENTATIONS ON THE TWO CONSIDERED GPUS

Dataset	Algorithm	Runtime ($\times 10^{-5}$ seconds)									
		GPU1 (GTX 1080)					GPU2 (TESLA P100)				
		CPU→GPU	Expectation	Theta	Lambda	GPU→CPU	CPU→GPU	Expectation	Theta	Lambda	GPU→CPU
Samson	pLSA	63.90	26.91	11.36	12.25	1.26	42.22	24.02	5.22	4.13	2.23
	DEpLSA	93.82	38.38	24.60	16.17	1.27	69.19	34.75	8.07	5.00	2.00
Jasper	pLSA	104.39	35.45	33.51	15.15	1.48	54.50	34.57	10.62	5.11	2.37
	DEpLSA	129.46	45.15	40.79	23.73	1.41	74.9	39.61	12.25	7.02	2.49
Urban	pLSA	726.08	323.28	530.55	129.90	4.97	253.84	232.65	126.93	31.92	6.65
	DEpLSA	-	-	-	-	-	444.63	352.73	214.05	60.04	3.39
Cuprite	pLSA	1114.99	317.66	1073.79	256.09	6.40	358.89	155.13	299.21	95.62	8.70
	DEpLSA	-	-	-	-	-	511.04	205.22	414.34	120.51	3.92

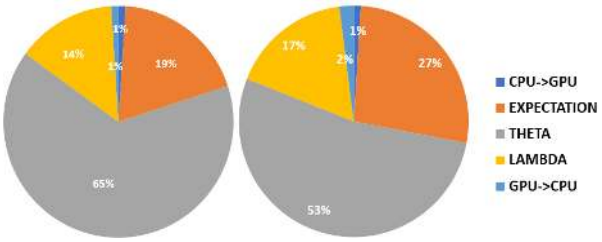


Fig. 8. Graphic diagram showing the percentage of time that each of the executed kernels consume on NVIDIA GeForce GTX 1080 (left) and NVIDIA Tesla P100 (right) when processing the Cuprite dataset. Is important to remark that I/O transfers are executed once, meanwhile kernels are executed iteratively, so the times in the diagrams have been weighted accordingly.

TABLE IV

PER-KERNEL OCCUPANCY (TOTAL CORES USAGE) WHEN ANALYZING THE CUPRITE DATASET ON GPU1.

Kernel name	Occupancy (%)
Theta	100
Lambda	94
Expectation	50

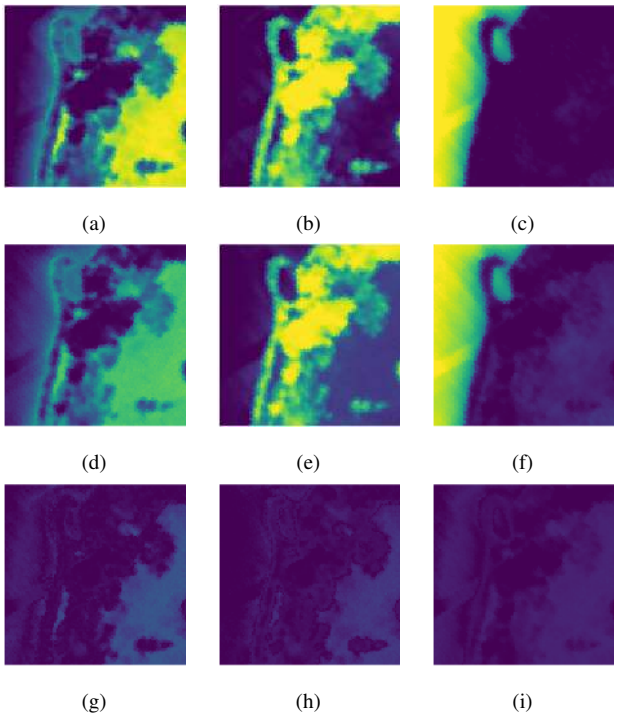


Fig. 9. Fractional abundance maps for the Samson dataset. (a)-(c) Ground truth abundances for each of the three endmembers. (d)-(f) Fractional abundance estimation for each of the three endmembers obtained by the DEpLSA algorithm (executed on the Tesla P100 GPU). (g)-(i) Absolute distances between estimated and real fractional abundances for each of the three considered endmembers, where dark colors indicate lower errors.

very close to the true ones in this particular scene.

In order to evaluate the computational performance of the GPU implementations, Table II shows the execution times (in seconds) for the serial versions (DEpLSA, pLSA), the optimized versions (OP-DEpLSA, OP-pLSA), and the parallel versions (implemented in both GPU1 and GPU2 architectures). The speedups achieved in these two GPU architectures are also displayed, together with the speedup obtained by the flag-

optimized versions with regards to the standard ones. As shown by Table II, the optimization via flags already provides some improvements in terms of computational time. However, it is the use of GPU architectures that leads to highly accelerated performance in all the cases. While the speedups obtained in the GPU1 architecture are around 30x (meaning that the code can be executed in the GPU at least 30 times faster), the speedups obtained in the GPU2 architecture can be up to 147x. These are quite significant acceleration factors. At this point, it is important to note that the times for DEpLSA in the GPU1 architecture could not be recorded for the Urban and Cuprite scenes, due to limitations in the video memory of the GPU.

For illustrative purposes, Fig. 10 displays graphically the speedups achieved by the GPU versions of pLSA and DEpLSA in the two considered GPU architectures: GTX 1080 (GPU1) and Tesla P100 (GPU2), for the different datasets considered in the experiments. As Fig. 10 shows, the achieved speedups are higher in the Tesla P100 architecture. This observation is related to the number of available cores (3584 in GPU2 versus 2560 in GPU1) as well as to the available video memory (16 GB in GPU2 versus 8 GB in GPU1). The fact that the Urban and Cuprite scenes cannot be processed in the GTX 1080 is also related to this difference in video memory between the two considered GPU architectures (8 GB vs 16 GB). By looking at the results in the Tesla P100 GPU, one can infer that the speedup increases with image size, which is a highly desirable feature given the increasing size and dimensionality of remotely sensed hyperspectral data repositories.

Also, to provide a visual and in-depth assessment of kernel performance, Table III illustrates how the kernels perform individually. It is important to emphasize that the transfers from the GPU to the CPU take more time in the GPU2 environment (due to a CPU bottleneck), since those CPUs are ARM-based and exhibit smaller bandwidth as compared with the GPU1 environment.

In order to test the robustness of our GPU implementation, it is also important to provide some in-depth performance indicators extracted from NVidia Visual Profiler. The obtained profiler data (see Fig. 8) confirm our introspection, explained in section III, that (due to the use of 3D computations instead of matrix computations), the Expectation kernel consumes most of the computing time, meanwhile the kernels devoted to computing the endmembers (#2 and #3) have minimal impact (i.e., kernel #1 kernel performs the majority of the computations). We also note that, as Table IV collects, our implementation takes advantage of almost all GPU cores the majority of execution time. It is important to remark that lower occupancies are not always related with lower performances.

V. Conclusions and Future Lines

In this work, we have introduced a new parallel version of the pLSA algorithm for efficiently conducting hyperspectral unmixing using the DepLSA model. Our newly developed implementation is able to run in a many-core specific platform (GPU). As a result, the presented approach provides an efficient and effective unmixing solution for actual remote sensing production environments.

Our experiments, conducted over four real hyperspectral datasets and two different GPU architectures, indicate that our many-core implementation takes full advantage of core-level parallelism, optimizing the heavy matrix computations involved in the process, achieving very similar results as the serial counterparts in terms of unmixing accuracy. It is also important to emphasize that our pLSA implementation fully exploits all the GPU capabilities, becoming more efficient with the latest-generation GPUs.

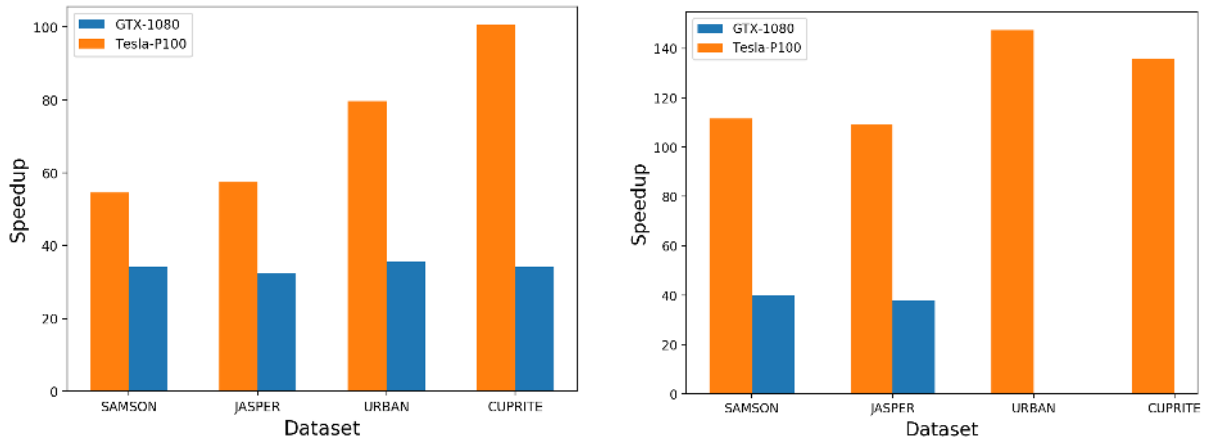


Fig. 10. Speedups achieved by the GPUpLSA (a) and GPUDEpLSA (b) regarding their serial versions (pLSA and DEpLSA, respectively), for the four different datasets considered in the experiments.

As with any new approach, there are some unresolved issues that may present challenges over time. In this sense, future lines will cover some relevant developments that were not included in the present study. Specifically, multi-GPU support may allow to decrease even more the computing time. Besides, considering a larger number of dimensions in the first step may help to optimize the DEpLSA results. Another future line worth being considered is to adopt other specific hardware accelerators, such as the Intel Xeon Phi or reconfigurable solutions like field programmable gate arrays (FPGAs), which are currently more suitable than GPUs for onboard exploitation [1], [51].

ACKNOWLEDGEMENT

The authors would like to take this opportunity to gratefully thank the Associate Editor and the Anonymous Reviewers for their outstanding comments and suggestions, that greatly helped us to improve the technical quality and presentation of our manuscript.

REFERENCES

[1] J. M. Bioucas-Dias, A. Plaza, G. Camps-Valls, P. Scheunders, N. Nasrabadi, and J. Chanussot, "Hyperspectral remote sensing data analysis and future challenges," *IEEE Geoscience and remote sensing magazine*, vol. 1, no. 2, pp. 6–36, 2013.

[2] P. Ghamisi, N. Yokoya, J. Li, W. Liao, S. Liu, J. Plaza, B. Rasti, and A. Plaza, "Advances in hyperspectral image and signal processing: A comprehensive overview of the state of the art," *IEEE Geoscience and Remote Sensing Magazine*, vol. 5, no. 4, pp. 37–78, 2017.

[3] J. Li, X. Zhao, Y. Li, Q. Du, B. Xi, and J. Hu, "Classification of hyperspectral imagery using a new fully convolutional neural network," *IEEE Geoscience and Remote Sensing Letters*, vol. 15, no. 2, pp. 292–296, 2018.

[4] M. Paoletti, J. Haut, J. Plaza, and A. Plaza, "A new deep convolutional neural network for fast hyperspectral image classification," *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 145, pp. 120 – 147, 2018, deep Learning RS Data. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0924271617303660

[5] M. E. Paoletti, J. M. Haut, R. Fernandez-Beltran, J. Plaza, A. J. Plaza, J. Li, and F. Pla, "Capsule Networks for Hyperspectral Image Classification," *IEEE Transactions on Geoscience and Remote Sensing*, no. 99, pp. 1–15, 2018.

[6] A. Ma, Y. Zhong, D. He, and L. Zhang, "Multiobjective subpixel land-cover mapping," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 56, no. 1, pp. 422–435, 2018.

[7] R. Fernandez-Beltran, J. M. Haut, M. E. Paoletti, J. Plaza, A. Plaza, and F. Pla, "Multimodal probabilistic latent semantic

analysis for sentinel-1 and sentinel-2 image fusion," *IEEE Geoscience and Remote Sensing Letters*, vol. 15, no. 9, pp. 1347–1351, 2018.

[8] D. Liu and L. Han, "Spectral curve shape matching using derivatives in hyperspectral images," *IEEE Geoscience and Remote Sensing Letters*, vol. 14, no. 4, pp. 504–508, 2017.

[9] N. Li, X. Huang, H. Zhao, X. Qiu, R. Geng, X. Jia, and D. Wang, "Multiparameter optimization for mineral mapping using hyperspectral imagery," *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 11, no. 4, pp. 1348–1357, 2018.

[10] J. M. Bioucas-Dias, A. Plaza, N. Dobigeon, M. Parente, Q. Du, P. Gader, and J. Chanussot, "Hyperspectral unmixing overview: Geometrical, statistical, and sparse regression-based approaches," *IEEE journal of selected topics in applied earth observations and remote sensing*, vol. 5, no. 2, pp. 354–379, 2012.

[11] W.-K. Ma, J. M. Bioucas-Dias, T.-H. Chan, N. Gillis, P. Gader, A. J. Plaza, A. Ambikapathi, and C.-Y. Chi, "A signal processing perspective on hyperspectral unmixing: Insights from remote sensing," *IEEE Signal Processing Magazine*, vol. 31, no. 1, pp. 67–81, 2014.

[12] J. M. Nascimento and J. M. Dias, "Vertex component analysis: A fast algorithm to unmix hyperspectral data," *IEEE transactions on Geoscience and Remote Sensing*, vol. 43, no. 4, pp. 898–910, 2005.

[13] J. Li, A. Agathos, D. Zaharie, J. M. Bioucas-Dias, A. Plaza, and X. Li, "Minimum volume simplex analysis: A fast algorithm for linear hyperspectral unmixing," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 53, no. 9, pp. 5067–5082, 2015.

[14] J. M. Nascimento and J. M. Bioucas-Dias, "Hyperspectral unmixing based on mixtures of dirichlet components," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 50, no. 3, pp. 863–878, 2012.

[15] A. Halimi, N. Dobigeon, J.-Y. Tourneret, and P. Honeine, "A new bayesian unmixing algorithm for hyperspectral images mitigating endmember variability," in *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2015, pp. 2469–2473.

[16] A. Huck and M. Guillaume, "Robust hyperspectral data unmixing with spatial and spectral regularized nmf," in *2010 2nd Workshop on Hyperspectral Image and Signal Processing: Evolution in Remote Sensing*. IEEE, 2010, pp. 1–4.

[17] J. Li, J. M. Bioucas-Dias, A. Plaza, and L. Liu, "Robust collaborative nonnegative matrix factorization for hyperspectral unmixing," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 54, no. 10, pp. 6076–6090, 2016.

[18] Y. Itoh, S. Feng, M. F. Duarte, and M. Parente, "Semisupervised endmember identification in nonlinear spectral mixtures via semantic representation," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 55, no. 6, pp. 3272–3286, 2017.

[19] R. Fernandez-Beltran, P. Latorre-Carmona, and F. Pla, "Latent topic-based super-resolution for remote sensing," *Remote Sensing Letters*, vol. 8, no. 6, pp. 498–507, 2017.

[20] R. Fernandez-Beltran, J. M. Haut, M. E. Paoletti, J. Plaza, A. Plaza, and F. Pla, "Remote sensing image fusion using hierarchical multimodal probabilistic latent semantic analysis," *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 11, no. 12, pp. 4982–4993, 2018.

[21] R. Fernandez-Beltran and F. Pla, "Sparse multi-modal probabilistic latent semantic analysis for single-image super-resolution," *Signal Processing*, vol. 152, pp. 227–237, 2018.

[22] ——, "Latent topics-based relevance feedback for video retrieval," *Pattern Recognition*, vol. 51, pp. 72–84, 2016.

[23] R. Fernandez-Beltran, A. Plaza, J. Plaza, and F. Pla, "Hyperspectral unmixing based on dual-depth sparse probabilistic latent semantic analysis," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 56, no. 11, pp. 6344–6360, 2018.

[24] R. Wan, V. N. Anh, and H. Mamitsuka, "Efficient probabilistic latent semantic analysis through parallelization," in *AIRS '09 Proceedings of the 5th Asia Information Retrieval Symposium on Information Retrieval Technology*, 2009, pp. 432–443.

[25] D. M. Chickering, "Learning bayesian networks is np-complete," in *Learning from data*. Springer, 1996, pp. 121–130.

[26] R. Fernandez-Beltran and F. Pla, "Incremental probabilistic latent semantic analysis for video retrieval," *Image and Vision Computing*, vol. 38, pp. 1–12, 2015.

[27] E. K. Kouassi, T. Amagasa, and H. Kitagawa, "Efficient probabilistic latent semantic indexing using graphics processing unit," *Procedia Computer Science*, vol. 4, pp. 382–391, 2011.

[28] Z. Liang, W. Li, and Y. Li, "A parallel probabilistic latent semantic analysis method on mapreduce platform," in *IEEE International Conference on Information and Automation (ICIA)*, 2013, pp. 1–10.

[29] Y. Ma, H. Wu, L. Wang, B. Huang, R. Ranjan, A. Zomaya, and W. Jie, "Remote sensing big data computing: Challenges and opportunities," *Future Generation Computer Systems*, vol. 51, pp. 47–60, 2015.

[30] S. Bernabe, S. Lopez, A. Plaza, and R. Sarmiento, "GPU implementation of an automatic target detection and classification algorithm for hyperspectral image analysis," *IEEE Geoscience and Remote Sensing Letters*, vol. 10, no. 2, pp. 221–225, March 2013.

[31] L. Santos, E. Magli, R. Vitulli, J. F. Lopez, and R. Sarmiento, "Highly-parallel GPU architecture for lossy hyperspectral image compression," *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 6, no. 2, pp. 670–681, April 2013.

[32] A. Agathos, J. Li, D. Petcu, and A. Plaza, "Multi-GPU implementation of the minimum volume simplex analysis algorithm for hyperspectral unmixing," *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 7, no. 6, pp. 2281–2296, June 2014.

[33] X. Li, B. Huang, and K. Zhao, "Massively parallel GPU design of automatic target generation process in hyperspectral imagery," *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 8, no. 6, pp. 2862–2869, June 2015.

[34] Z. Wu, J. Liu, A. Plaza, J. Li, and Z. Wei, "GPU implementation of composite kernels for hyperspectral image classification," *IEEE Geoscience and Remote Sensing Letters*, vol. 12, no. 9, pp. 1973–1977, Sep. 2015.

[35] E. M. Sigurdsson, A. Plaza, and J. A. Benediktsson, "GPU implementation of iterative-constrained endmember extraction from remotely sensed hyperspectral images," *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 8, no. 6, pp. 2939–2949, June 2015.

[36] K. Tan, J. Zhang, Q. Du, and X. Wang, "GPU parallel implementation of support vector machines for hyperspectral image classification," *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 8, no. 10, pp. 4647–4656, Oct 2015.

[37] E. Torti, G. Danese, F. Leporati, and A. Plaza, "A hybrid CPU-GPU real-time hyperspectral unmixing chain," *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 9, no. 2, pp. 945–951, Feb 2016.

[38] J. Sevilla, G. Martin, and J. M. P. Nascimento, "Parallel hyperspectral unmixing method via split augmented lagrangian on GPU," *IEEE Geoscience and Remote Sensing Letters*, vol. 13, no. 5, pp. 626–630, May 2016.

[39] X. Wu, B. Huang, L. Wang, and J. Zhang, "GPU-based parallel design of the hyperspectral signal subspace identification by minimum error (hysime)," *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 9, no. 9, pp. 4400–4406, Sep. 2016.

[40] L. I. Jimenez, G. Martin, S. Sanchez, C. Garcia, S. Bernabe, J. Plaza, and A. Plaza, "GPU implementation of spatialspectral preprocessing for hyperspectral unmixing," *IEEE Geoscience and Remote Sensing Letters*, vol. 13, no. 11, pp. 1671–1675, Nov 2016.

[41] J. Lopez-Fandino, B. Priego, D. B. Heras, and F. Arguello, "GPU projection of ECAS-II segmenter for hyperspectral images based on cellular automata," *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 10, no. 1, pp. 20–28, Jan 2017.

[42] E. Martel, R. Guerra, S. Lopez, and R. Sarmiento, "A GPU-based processing chain for linearly unmixing hyperspectral images," *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 10, no. 3, pp. 818–834, March 2017.

[43] W. Li, L. Zhang, L. Zhang, and B. Du, "GPU parallel implementation of isometric mapping for hyperspectral classification," *IEEE Geoscience and Remote Sensing Letters*, vol. 14, no. 9, pp. 1532–1536, Sep. 2017.

[44] A. Ordonez, F. Arguello, and D. B. Heras, "GPU accelerated FFT-based registration of hyperspectral scenes," *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 10, no. 11, pp. 4869–4878, Nov 2017.

[45] Z. Wu, L. Shi, J. Li, Q. Wang, L. Sun, Z. Wei, J. Plaza, and A. Plaza, "GPU parallel implementation of spatially adaptive hyperspectral image classification," *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 11, no. 4, pp. 1131–1143, April 2018.

[46] S. Bernabe, S. Sanchez, A. Plaza, S. Lopez, J. A. Benediktsson, and R. Sarmiento, "Hyperspectral unmixing on GPUs and multi-core processors: A comparison," *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 6, no. 3, pp. 1386–1398, June 2013.

[47] J. M. P. Nascimento, J. M. Bioucas-Dias, J. M. Rodriguez Alves, V. Silva, and A. Plaza, "Parallel hyperspectral unmixing on GPUs," *IEEE Geoscience and Remote Sensing Letters*, vol. 11, no. 3, pp. 666–670, March 2014.

[48] D. M. Blei, "Probabilistic topic models," *Communications ACM*, vol. 55, no. 4, pp. 77–84, 2012.

[49] R. Fernandez-Beltran and F. Pla, "Prior-based probabilistic latent semantic analysis for multimedia retrieval," *Multimedia Tools and Applications*, vol. 77, no. 13, pp. 16 771–16 793, 2018.

[50] F. Zhu, Y. Wang, B. Fan, S. Xiang, G. Meng, and C. Pan, "Spectral unmixing via data-guided sparsity," *IEEE Transactions on Image Processing*, vol. 23, no. 12, pp. 5412–5427, 2014.

[51] J. M. Haut, S. Bernab, M. E. Paoletti, R. Fernandez-Beltran, A. Plaza, and J. Plaza, "Low-high-power consumption architectures for deep-learning models applied to hyperspectral image classification," *IEEE Geoscience and Remote Sensing Letters*, pp. 1–5, 2018.